02

¿Herencia múltiple?

Transcripción

[00:00] Sean bienvenidos a una nueva clase de su curso de Java. Ya a estas alturas estamos digamos con conocimiento bien avanzado, incluso yo podría decir que ya diferenciaba a nivel ya de los cursos anteriores. ¿Por qué? Porque ya hemos explorado conceptos que son utilizados incluso en el mismo lenguaje Java como son la herencia y el polimorfismo.

[00:29] Vamos a poner un poco más complejo nuestro caso de uso, quiero decir vamos a agregar una complejidad más a los ejemplos que estamos dando. Sabemos que hasta ahora, si abrimos nuestro proyecto de Byte Bank heredado que hemos creado aquí, tenemos muy pocas clases por así decirlo. Aún no tenemos una complejidad tan grande.

[00:58] Tenemos clase cuenta, tenemos un contador, funcionario, gerente, en fin. Básicamente si vamos a nuestro diagrama de clases, en este momento lo que nosotros tenemos es algo así.

[01:12] Tenemos un funcionario que tanto de gerente como de contador extienden del funcionario, entonces esto permite reutilizar el código en algunas formas, por ejemplo lo que permite el polimorfismo es que si nosotros necesitamos un método que acepte tanto gerente como contador, entonces simplemente mandamos un funcionario como parámetro y el método ya va ha digamos asumir que puede ser un gerente o un contador. Perfecto.

[01:51] Ahora vamos a adicionar un requerimiento más. Digamos que nuestro Byte Bank está creciendo y a medida que crece el sistema también necesita

crecer, el sistema crece junto con el negocio, entonces ahora no solamente vamos a tener gerentes y contadores sino que va a entrar un tercer personaje que va a ser nuestro administrador del sistema. Entonces, vamos a representarlo en nuestro diagrama, aquí a este nivel, y le vamos a poner administrador.

[02:26] Perfecto. Y como ya hemos visto antes, entonces el administrador también va a extender de funcionario, porque el administrador también es un funcionario. Ya hemos visto que a medida que vamos agregando más personajes, más actores a nuestro sistema, si nosotros necesitamos representarlo, una orientación a objetos básica, entonces diríamos que el administrador extiende del funcionario.

[03:00] Ahora, dentro de estos requerimientos, como el sistema mismo está creciendo, ¿qué sucede? Recuerdan que el gerente podía hacer login. Ahora el administrador también va a poder hacer login también. También va a poder autenticarse en el sistema. Entonces, desde ahora tanto gerente como administrador son capaces de autenticarse en el sistema.

[03:25] Entonces vamos a ver un poco el código, primero vamos a implementar a nivel de código al administrador, para ver más o menos cómo podría quedar esto. Para eso, ustedes ya saben el proceso para crear una nueva clase, damos new class, y aquí le damos Administrador. Recuerden siempre comenzamos con mayúscula porque estamos refiriéndonos a un objeto o entidad administrador.

[03:52] Y le decimos que la superclass, vamos a darle browse, la superclass no va a ser object sino va a ser funcionario. Hasta ahora solamente estamos imitando el mismo comportamiento que teníamos anteriormente. Le damos finish y él ya automáticamente genera el código. Ahora vamos a la clase del gerente.

[04:13] Aquí en la clase del gerente, aparte del método getBonificacion que tenemos ya aquí en el administrador implementado que retorna a cero, voy a cerrar las demás clases, solamente para tener un poco más de orden aquí. Esto es una costumbre que yo siempre tengo en las clases que no uso siempre la cierro para que no me ocupen digamos el espacio aquí y sea mucho más rápido para mí encontrar las clases que de verdad necesito.

[04:43] El contador está exactamente igual también, él solamente tiene bonificación. ¿Por qué? Porque el contador no autentica, no tiene contraseña. Y si nos damos cuenta, aquí tenemos pues clave un método setClave y un método iniciarSesion. Perfecto. Hasta aquí todo tranquilo. Ahora vamos a crear pues nuestra clase que va a gerenciar digamos estas autenticaciones, estos logins en el sistema.

[05:15] Para eso lo que vamos a hacer es un new, nuevamente class, y la clase se va a llamar Sistema Interno, para representar pues nuestro propio sistema que tenemos dentro del banco, dentro del Byte Bank. Le damos finish, y en SistemaInterno obviamente vamos a crear nuestro método autentica, que va a ser un booleano. ¿Por qué? Porque nos va a retornar si puedo autenticar o no puedo autenticar.

[05:48] Entonces vamos a ponerle autentica, perfecto. Return por defecto false para que no nos deje con el error ahí, y listo. Entonces, ¿yo qué debería evaluar aquí? Si yo quiero que mi sistema interno autentique a alguien, digamos que en este caso sigo con el gerente, porque el gerente se autentica, entonces le voy a poner como parámetro gerente. Gerente gerente. Perfecto.

[06:20] ¿Y qué es lo que yo voy a retornar aquí? Aquí lo que yo voy a retornar es el método del gerente IniciarSesion, pero para iniciar sesión necesito una clave, entonces aquí digamos que yo lo que voy a hacer es decir un booleano, boolean puedeIniciarSesion. ¿Va a ser igual a qué? A gerente.iniciarSesion ¿con qué clave? Con una que voy a definir aquí en este momento.

[07:05] Y esa clave va a ser un private int Clave, que va a ser igual a 12345, punto y coma para que compile, y le debería enviar la clave. Entonces, aquí una clave como parámetro. Y listo. Vemos que ya está dando un error aquí. ¿Por qué? Porque iniciar sesión yo creo que acepta un string. Así es, acepta un string, no un entero.

[07:40] Entonces vamos a complacerlo por esta vez y lo que hacemos es esto lo volvemos un string. Listo. No va a haber diferencia aquí. Y al final es un punto y coma. Entonces, digamos, si, ya he creado mi cuerpo del if, si puede iniciar sesión entonces vamos a imprimirle algún mensaje, que va a ser: "Login exitoso". Perfecto.

[08:22] Si no: "Error en login". Entonces, ya tenemos nuestro método autentica. Perdón, y de aquí obviamente nos está dando error aquí. ¿Por qué? Porque tenemos que retornar true, porque pudo autenticar o return false porque no pudo autenticar. Ahora sí. Ya tenemos nuestro método autentica en la clase gerente. Perfecto.

[08:59] Ahora, ya hemos quedado que el administrador también puede autenticar. ¿Entonces qué hacemos? Como el administrador también es un funcionario, también puede autenticar, el gerente ya tiene los atributos para autenticar que son la clave, el método para setear la clave y el método para iniciar sesión. Entonces, aquí lo que vamos a hacer es copiar este código. Perfecto.

[09:27] Le damos "Ctrl + C" y aquí, voy a cerrar el contador, en el administrador ponemos un espacio aquí y "Ctrl + V". ¿De esta forma qué estamos haciendo? Estamos duplicando la funcionalidad del método iniciar sesión. Y si ustedes son curiosos, si ustedes ya están viendo aquí, se dan cuenta que algo ya comienza pues a oler mal, alguna cosa aquí no está bien. Pero bueno, vamos a dejarlo así.

[09:57] Volvemos aquí a nuestro sistema interno. Y ahora necesitamos hacer que el administrador también se autentique, pero acá volvemos a un problema que hemos visto si no me equivoco hace dos clases. ¿Cuál es? El tipo de parámetro que está yendo aquí. ¿Por qué? Porque si yo quiero que ahora el administrador también inicie sesión, entonces, yo tendría que crear otro método autentica, digamos copiar todo esto, hacer "Ctrl + C" y aquí "Ctrl + V", pero la del administrador.

[10:38] ¿Cuál es el problema aquí? Ese problema ya lo hemos visto, ¿se dan cuenta que estamos duplicando el código? Pero ahora ustedes me dirán: "Pero Diego, calma. ¿Por qué te complicas la vida? Puedes tranquilamente borrar todo esto y aquí envías un funcionario". Envías un funcionario. Ahora, aquí hay dos detalles.

[11:05] ¿Cuál es el primer detalle? Funcionario no tiene el método iniciarSesion implementado. ¿Se soluciona? Se soluciona, lo implementamos. ¿Y cuál es el segundo detalle, y yo creo que es el más importante? Volvemos a nuestro diagrama.

[11:22] Ya que funcionario, ahora puede iniciar sesión y contador, gerente y administrador extienden del funcionario, entonces ¿no creen ustedes que habilitaríamos el inicio de sesión tanto para el administrador, el gerente? Okay, ellos dos satisfacen nuestra necesidad. ¿Y él? Él también tendría habilidada esa funcionalidad, y nosotros no queremos que la tenga.

[11:53] Ahí ya comenzamos a ver digamos un problema de diseño. Vamos a ver en el próximo video cómo es que podemos afrontar este problema y una solución muy ingeniosa y yo estoy seguro que les va a servir en más de una ocasión.